

ICBMC: An Improved Cohesion Measure for Classes

Yuming Zhou Baowen Xu

Department of computer Science & Engineering, Southeast University, Nanjing, China

National Key Laboratory of Software Engineering, Wuhan University, Wuhan, China

Jianjun Zhao

Department of Computer Science & Engineering, Fukuoka Institute of Technology, Japan

Hongji Yang

Department of Computer Science, De Montfort University, England

Abstract

Class cohesion could be used to evaluate the design quality of classes, to develop test measures for object-oriented software and to restructure poorly designed classes. Among a number of class cohesion measures proposed in the last decade, H. S. Chae's measure is based on the structure of the reference graph of a class, which overcomes the limitations of most class cohesion measures. However, it only considers the patterns of interactions among the members of a class partly and hence does not satisfy monotonicity, which might cause the measuring results inconsistent with intuition in some cases. This paper first analyzes the limitations of typical cohesion measures for classes in detail, and then proposes an improved cohesion measure ICBMC. Finally, this paper exemplifies the advantages and applications of ICBMC.

1. Introduction

Cohesion, a measure of the degree to which the elements of a module belong together, was first introduced by Stevens^[1]. In the last three decades, a large number of module cohesion measures have been proposed. Subsequently, the concept was migrated to object-oriented systems. Classes in object-oriented systems group data and related operations belonging to a specific domain concept, which support object-oriented features such as data abstraction, encapsulation and inheritance. Currently, the principle that object-oriented classes should have high cohesion has been widely accepted. That is to say, the higher the cohesion of a class is, the easier the class is to be developed, maintained, and reused. Because of not considering these object-oriented features, previous module cohesion measures could not be used to measure

the cohesiveness of classes directly. Many cohesion measures for classes are hence developed to depict this complex software attribute^[2-7].

However, the lacking of theory foundation of these metrics makes practitioners difficult to compare and evaluate them, i.e., it is unclear that which is valid and which is invalid. As a consequence, practitioners do not know what cohesion metric is most suitable to measure the cohesion of classes in object-oriented systems. Therefore, Briand proposed four cohesion properties—nonnegativity and normalization, null value and maximum value, monotonicity, and merging of unconnected classes—and pointed out that a valid cohesion metric should satisfy them. Practitioners can use these four cohesion properties to compare and evaluate existing cohesion metrics and thereby can know how to select a suitable cohesion metric based on a particular goal of measurement^[8]. Briand gave the reason why a cohesion metric should satisfy these cohesion properties as follows:

“The motivation behind defining such properties is that a measure must be supported by some underlying theory - if it is not, then the usefulness of that measure is questionable. The four cohesion properties defined by Briand et al. are one of the more recent proposals to characterize cohesion in a reasonably intuitive and rigorous manner. While these properties are not sufficient to say that a measure which fulfils them all will be useful, it is likely that a measure which does not fulfil them all is ill-defined.”

Most class cohesion measures, however, do not consider the nature of classes, which might cause that the measuring value of cohesion of a class is inconsistent with the real value of its cohesion. On the other hand, those measures take account of only the number of interactions, not the patterns of interactions among the members of a class, which might draw a conclusion inconsistent with

This work was supported in part by the National Natural Science Foundation of China (NSFC) (60073012), Natural Science Foundation of Jiangsu, China (BK2001004), Opening Foundation of State Key Laboratory of Software Engineering in Wuhan University, Foundation of State Key Laboratory for Novel Software Technology in Nanjing University, Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology, and Visiting Scholar Foundation of Key Lab. in University.

intuition. To overcome above-mentioned limitations, H. S. Chae proposes a new cohesion measure CBMC for classes [6,7]. In fact, this measure uses a reference graph to present the access relations between the methods and instance variables of a class, decomposes the structure of the reference graph by the set of glue methods, and then defines its cohesion as the product of the connectivity factor and the structure factor.

Chae shows that the new class cohesion measure has many advantages over previous measures and uses it to evaluate the design quality of classes and reconstruct poorly designed classes in object-oriented systems. However, when Briand's four cohesion properties are used to verify Chae's cohesion metric, it is easy to know that CBMC does not fulfill monotonicity. That is to say, it might cause incorrect measuring results in some cases. This means that CBMC is ill defined, which limits its application in the evaluation of design quality of classes. The reason is that CBMC only partly considers the patterns of interactions among the members of a class [9-10].

The remainder of this paper is organized as follows. Section 2 discusses and analyzes the limitations of typical cohesion measures for classes in detail. Then, section 3 proposes a new class cohesion measure, ICBMC, which actually is an improved CBMC. Section 4 shows that ICBMC has many advantages over existing typical cohesion measures for classes and section 5 exemplifies how ICBMC could be used to restructure poorly designed classes. Finally, conclusions are given in section 6.

2. Limitations of typical cohesion measures

In the last decade, a large number of cohesion measures for classes in object-oriented systems have been proposed. Briand performed a comprehensive review of these class cohesion measures by the theoretical validation of four cohesion properties. As a result, Briand defined a new framework for cohesion measurement, which make practitioners know how to select a most suitable cohesion measure for a particular measurement goal and provide researchers guidelines to develop new measures [8]. Since then, more cohesion measures for classes have been proposed. Although some cohesion measures satisfy four cohesion properties, there does not exist one measure whose measurement is completely consistent with intuition. In the following, we will analyze the limitations of typical cohesion measures for classes.

2.1. LCOM by Chidamber [2]

Chidamber and Kemerer defined cohesion of a class based on similarity of its methods. Suppose $M(c)$ be the set of methods defined in class c and $\lambda(m, c)$ the set of instance variables accessed by a method m in class c . For

any two different methods m_i and m_j in class c , if $\lambda(m_i, c) \cap \lambda(m_j, c) \neq \emptyset$, then it can be claimed that m_i and m_j are similar, formally denoted by $Similar(m_i, m_j, c)$. Therefore, Chidamber's cohesion measure can be described as follows:

$$LCOM = \begin{cases} |P| - |Q|, & \text{if } |P| > |Q| \\ 0, & \text{otherwise} \end{cases}$$

where

$$P = \begin{cases} \emptyset, & \text{if } \bigcup_{m_i \in M(c)} \lambda(m_i, c) = \emptyset \\ \{(m_i, m_j) \mid \neg Similar(m_i, m_j, c)\}, & \text{otherwise} \end{cases}$$

$$Q = \{(m_i, m_j) \mid Similar(m_i, m_j, c)\}$$

It is easy to conclude that LCOM1 does not satisfy normalization and monotonicity properties. Therefore, LCOM might draw a conclusion inconsistent with intuition in most cases. For example, consider class b1 and b2 shown in figure 1. According to Chidamber's cohesion measure, $LCOM(b1) = 1 - 0 = 1$ and $LCOM(b2) = 6 - 4 = 2$. That is to say, it could be concluded that class b1 more cohesive than class b2. Obviously, this conclusion is inconsistent with our intuition.

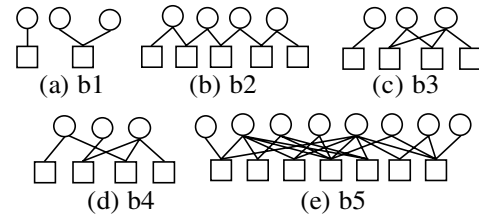


Figure 1 some classes

2.2. RCI by Briand [8]

Briand believes that a class consists of data declaration and methods. For two data declarations a and b , a DD-interaction b only if a change in a 's declaration or use may cause the need for a change in b 's declaration or use. For data declaration a and method m , a DM-interaction m only if a DD-interaction with at least one data declaration of m . Based on these notions, Briand defined cohesion measure as follows:

$$RCI(c) = \frac{|CI(c)|}{Max(c)}$$

where $CI(c)$ is the set of all DD- and DM-interactions and $Max(c)$ is the set of all possible DD- and DM-interactions of class c .

For a given class c , $RCI(c)$ is the ratio of the number of actual interaction to that of possible interaction. Although RCI satisfy all four cohesion properties, it might draw a conclusion that is inconsistent with intuition. For example, consider class b3 and class b4 provided by Chae [6,7](shown in figure 1). According to Briand's cohesion measure, they have the same cohesion value 6/12.

However, the interaction graph of class b3 is connected but that of class b4 is not, i.e., class b3 is more cohesive than class b4 with intuition.

2.3. OL_n by Yang^[11]

Yang believes that both of methods and instance variables contribute to the cohesion of a class. Therefore, $COM(m)$ and $COV(v)$ are used to represent the strength of method m and instance variable v respectively in Yang's cohesion measure. For a given class c , Yang first constructs a corresponding reference graph G_c . If G_c is disjoint, then let its cohesion be 0, otherwise computes its cohesion value by the following steps:

- (1) Initialize the iteration count variable $count$ to 0, i.e., let $count=0$;
- (2) For every method m_i , set its initial strength to 1, i.e., let $COM(m_i)=1$;
- (3) For every instance variable v_i , let $COV(v_i)=\frac{1}{N_m(G_c)} \sum_{j=1}^l COM(m_j)$, where $N_m(G_c)$ denotes the number of method vertexes in G_c and $\lambda(m_j, c)$;
- (4) For every method m_i , let $COM(m_i) = \frac{1}{N_v(G_c)} \sum_{k=1}^m COV(v_k)$, where $N_v(G_c)$ denotes the number of instance variable vertexes in G_c and $v_k \in \lambda(m_i, c)$;
- (5) $count = count + 1$, if $count < n$ then Go to (3)
- (6) Compute the cohesion of class c , i.e., let

$$OL_n(c) = \frac{1}{N_v(G)} \sum_{r=1}^{N_v(G)} COV(v_r).$$

It is easy to see that the computation of Yang's cohesion measure is an iteration process. Obviously, Yang's OL_n satisfies all of four cohesion properties. For a connected graph G_c , OL_1 is completely equivalent to Briand's RCI. Although OL_n has advantage over RCI, there exist some limitations in Yang's cohesion measure. First, it does not provide practitioners guidelines to select a suitable iteration factor n . Therefore, it is difficult to apply this cohesion measure in practice. Second, it might cause that $OL_{n-1}(c1) > OL_{n-1}(c2)$ but $OL_n(c1) < OL_n(c2)$. That is to say, it might draw a self-contradictory conclusion. For example, consider class b5 shown in figure 1(e) and class c6 shown in figure 5(a). It is easy to know that $OL_1(b5)=0.464286 > OL_1(c6)=0.392857$ but $OL_8(b5)=0.000013 < OL_8(c6)=0.000014$. Third, OL_n might draw a conclusion that is inconsistent with intuition. This problem will be further discussed in section 4.

2.4. CBMC by Chae^[6,7]

For a given class c , Chae uses a reference graph G_c to represent the access relations among its methods and instance variables, which is defined as

$$G_c = (N_c, E_c)$$

where

$$N_c = M(c) \cup V(c)$$

$$E_c = \{ \langle m, v \rangle \mid v \in V(c) \wedge m \in NM(c) \wedge v \in \lambda(m, c) \}$$

$M(c)$ and $V(c)$ represent the set of methods and the set of instance variables of class c respectively, and $NM(c)$ denotes the set of normal methods of class c ^[6-7].

For a reference graph G_c , the set of glue methods $M_g(G_c)$ is a set of normal methods of class c that satisfies the following two characters:

- (1) Without the members of $M_g(G_c)$ and related edges, the reference graph G_c becomes disjoint.
- (2) $M_g(G_c)$ is the minimum set of normal methods that holds character 1.

When all methods of $M_g(G_c)$ and related edges are removed, each connected sub graph is called a cohesion component (CC). If each method has interactions with all instance variables in a CC, then the CC is a most cohesive component (MCC). The cohesion of a MCC is defined as 1 in Chae's method.

Chae convinces that the cohesion of a class depends on not only the connectivity of itself, but also the cohesion of its constitute components. Let a reference graph G_c be the parent node, let its constitute components child nodes and apply the rule to those child nodes recursively, then the structure tree of G_c can be constructed. Because a reference graph can be separated by different sets of glue methods, the decomposition of a reference graph is not always unique. In that case, the set of glue methods, which results in a higher cohesion for the children, is selected. After above process, a reference graph G corresponds to a unique structure graph, whose cohesion is defined as:

$$CBMC(G) = F_c(G) \times F_s(G)$$

where

$$F_c(G) = \frac{|M_g(G)|}{|N_m(G)|} \quad F_s(G) = \frac{1}{n} \times \sum_{i=1}^n CBMC(G^i)$$

The connectivity factor $F_c(G)$ represents the strength of the connectivity among the members of class c and the structure factor $F_s(G)$ denotes the degree of the contribution of the components to the cohesion of class c . Therefore, the cohesion of a class c is defined to be

$$Cohesion(c) = CBMC(G_c)$$

Obviously, class cohesion is related to not only the number of interactions, but also the patterns of interactions among the constitute components of a class. The more complex the interactions among the members of a class are, the more cohesive the class is. Currently, many class cohesion measures have been proposed, most of which consider only the interaction number, not the interaction patterns. Chae convinces that the cohesion of a class is related to not only the strength of the connectivity of itself, but also the cohesion of

its constitute components. Therefore, Chae's measure represents the strength of the connectivity of itself by the connectivity factor, and presents the contribution of its constitute components to the class cohesion by the structure factor. Although Chae's measure overcomes the limitations of previous class cohesion measures to some extent, there exist some problems that cause the measuring value of class cohesion inconsistent with our intuition in the following respects.

(1) **A class with complex interactions might have the same measuring value of cohesion with a class with simple interactions.** Obviously, the more complex the interaction of the reference graph of a class is, the higher the cohesion of the class has. However, when adding an interaction edge to a reference graph, if the number of its constitute components decrease and the high of the structure trees remains no change, then the cohesion of the class might remains no change. For example, the reference graphs of class c1 and c3 are shown in figure 2(a) and 2(c) respectively, and figure 3(a) and 3(c) depict the corresponding structure trees, in which the set of glue methods consists of methods with bold line. Although both class c1 and c3 have the same methods and instance variables, the interactions on G_{c3} are more complex than that of on G_{c1} due to the fact that G_{c3} have more interaction edges than G_{c1} . Therefore, class c3 should be more cohesive than class c1 in intuition.

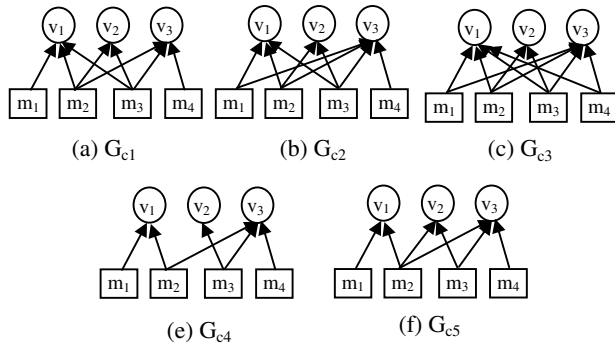


Figure 2 The reference graphs for c1, c2, c3, c4 and c5

However, according to the definition of Chae's measure, it is easy to conclude that

$$\begin{aligned} CBMC(G_{c1}) &= F_c(G_{c1}) \times F_s(G_{c1}) \\ &= F_c(G_{c1}) \times \frac{1}{3} \times [CBMC(G_{c1}^1) + CBMC(G_{c1}^2) + CBMC(G_{c1}^3)] \\ &= \frac{2}{4} \times \frac{1}{3} \times (1 + 1 + 1) = 1/2 \end{aligned}$$

Obviously, $CBMC(G_{c3}) = 1/2$. Therefore, class c1 and c3 have the same cohesion value, which is not consistent with our intuition. On the other hand, when adding an interaction edge to a reference graph, if both the number of its constitute components and the high of the structure trees remain no change, then Chae's measure also does not discriminate their cohesiveness. For instance, figure 2(e) and figure 2(f) show the reference graphs of class c4 and

c5 respectively, in which G_{c5} has more interactions than G_{c4} . Therefore, it can be claimed that class c5 is more cohesive than class c4. According to Chae's measure, they are said have the same cohesion value 3/16.

(2) **A class with complex interactions might have a lower measuring value of cohesion than a class with simple interactions.** When adding an interaction edge to a reference graph, if the number of its constitute components decreases and the high of the structure trees increases, then the cohesion value measured by Chae's measure might decrease to some extent.

For example, for class c1 and c2 shown in figure 2, although they have the same constitute components, class c2 is more cohesive than class c1 due to the fact that the interactions on G_{c2} is more complex than that on G_{c1} . According to Chae's measure, it is easy to know that

$$\begin{aligned} CBMC(G_{c2}) &= F_c(G_{c2}) \times F_s(G_{c2}) \\ &= F_c(G_{c2}) \times \frac{1}{2} \times [CBMC(G_{c2}^1) + CBMC(G_{c2}^2)] \\ &= F_c(G_{c2}) \times \frac{1}{2} \times \{F_c(G_{c2}^1) \times \frac{1}{2} \times \\ &\quad [CBMC(G_{c2}^{11}) + CBMC(G_{c2}^{12})] + CBMC(G_{c2}^2)\} \\ &= 3/8 \end{aligned}$$

Therefore, a conclusion that the cohesion of class c2 is lower than that of class c1 can be drawn, which is conflicted with our intuition.

Consequently, it can be concluded that Chae's measure is not suitable to evaluate the cohesion for classes in object-oriented systems in many cases, which limits its application to quality evaluation, software test and program understanding. These two above-mentioned cases also lead to the following self-contradictory problems in Chae's measure. In some cases, the more complex the interactions among the members of a class are, the lower its cohesion is (e.g., the changes from class c1 to c2). In other cases, the more complex the interactions among the members of a class are, the higher its cohesion is (e.g., the changes from class c2 to c3).

3. ICBMC — An improved class cohesion measure

Among these typical cohesion measures for classes discussed in section 2, it is not difficult to see that Chae's CBMC have some advantages over other measures (because CBMC partly considers the pattern of interactions). At the same time, it also can be seen that CBMC has the problems that lead to the inconsistent cases with our intuition. Therefore, we will propose an improved class cohesion measure based on CBMC in this section.

The reason why CBMC might draw a conclusion inconsistent with intuition is that the set of glue methods is used to decompose the reference graph of a class in Chae's measure, which could not represent the patterns of the interactions. To overcome these limitations, it is better to use a cut set, other than the set of glue methods, to decompose the

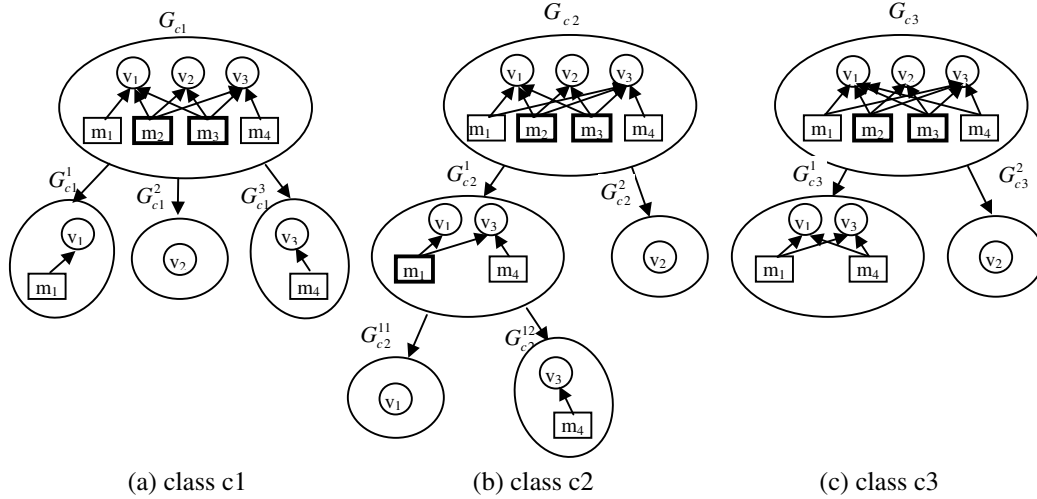


Figure 3 The structure trees of G_{c1} , G_{c2} and G_{c3}

reference graph. Therefore, the definitions of elementary components and non-elementary components are first given as follows.

Definition 1 Elementary components and non-elementary components Given the reference graph, $G_c = (N_c, E_c)$, of a class c , if a graph $G = (N, E)$ satisfies

- (1) $N \subseteq N_c$ and $E \subseteq E_c$
- (2) $\rho(G) = 1$ and $E \neq \emptyset$
- (3) $|N_m| = 1$ or $|N_v| = 1$ (where N_m and N_v denote the set of vertexes representing methods and the set of vertexes representing instance variables of class c , respectively)

then G is called an elementary component of G_c . If a graph $G' = (N', E')$ is a subgraph of the reference graph G_c and its non-spanning graphs contain elementary components, then G' is called a non-elementary component of G_c .

Definition 1 states that an elementary component is a connected subgraph of a reference graph, which contains at least two vertexes but only one method vertex or instance variable vertex. The elementary components and the non-elementary components are called the components of the reference graph of a class.

There are many approaches to decompose the reference graph of a class, but not all of them have good characters. In Chae's cohesion measure, the set of glue methods is used to decompose the reference graph of a class. When the interactions increase, the decomposition technique of the set of glue methods cannot assert that the cohesion value increases. We believe that a cut set of the reference graph of a class is very suitable to decompose its structure. A cut set of a reference graph is defined as follows.

Definition 2 The cut set of a reference graph Let $G_c = (N_c, E_c)$ is the reference graph of a class c

- (a) If $\rho(G_c) = 1$, then a cut set of G_c is the set of edges, Q , satisfies:
 - (1) G_c can be decomposed into two components if all edges in Q are removed.
 - (2) Q is the minimum set of edges, i.e. any proper

subset of Q does not hold character 1.

- (b) If $\rho(G_c) > 1$, then $Q = \emptyset$.

Let $CS(G_c)$ be the set of all cut sets of the reference graph G_c , i.e. $CS(G_c) = \{ Q \mid Q \text{ is a cut set of } G_c \}$. For the

reference graphs shown in figure 2, it is easy to know that

$$CS(G_{c1}) = \{ \{ \langle M_2, V_1 \rangle, \langle M_3, V_1 \rangle \}, \{ \langle M_2, V_3 \rangle, \langle M_3, V_3 \rangle \} \}$$

$$CS(G_{c2}) = \{ \{ \langle M_1, V_3 \rangle, \langle M_2, V_1 \rangle, \langle M_3, V_1 \rangle \},$$

$$\{ \langle M_1, V_3 \rangle, \langle M_2, V_3 \rangle, \langle M_3, V_3 \rangle \},$$

$$\{ \langle M_2, V_1 \rangle, \langle M_2, V_3 \rangle, \langle M_3, V_2 \rangle \},$$

$$\{ \langle M_2, V_2 \rangle, \langle M_3, V_1 \rangle, \langle M_3, V_3 \rangle \} \}$$

$$CS(G_{c3}) = \{ \{ \langle M_2, V_1 \rangle, \langle M_2, V_3 \rangle, \langle M_3, V_2 \rangle \},$$

$$\{ \langle M_2, V_2 \rangle, \langle M_3, V_1 \rangle, \langle M_3, V_3 \rangle \} \}$$

$$CS(G_{c4}) = \{ \{ \langle M_2, V_1 \rangle \}, \{ \langle M_2, V_3 \rangle \}, \{ \langle M_3, V_3 \rangle \} \}$$

$$CS(G_{c5}) = \{ \{ \langle M_2, V_1 \rangle \} \}$$

Let $\overline{G_Q}$ be the non-connected graph obtained from removing all edges in Q , each connected subgraph of $\overline{G_Q}$ be G_Q^i ($1 \leq i \leq \rho(\overline{G_Q})$), then the structure tree can be redefined as

Definition 3 Structure tree The structure tree for a reference graph $G_c = (N_c, E_c)$, $T_s(G_c)$, is a tree $T = (N, A)$ with the root node $(G_c, Q(G_c))$, where

$$(1) N = \{ (G_c^i, Q(G_c^i)) \mid G_c^i \subseteq G_c \}$$

$$(2) A = \{ ((G_c^p, Q(G_c^p)), (G_c^s, Q(G_c^s))) \mid G_c^s \in \overline{G_{cQ}^p} \}$$

Obviously, $\rho(\overline{G_Q}) = 2$. Therefore, if the reference graph of a class is a connected graph, then the corresponding structure tree must be a binary tree. Because a reference graph might have many cut sets, the decomposition of a reference graph is not always unique, i.e. there might exist many structure trees corresponding to a reference graph. The reference graph G_{c1} shown in figure 2 corresponds to six structure trees, three of which are depicted in figure 4. It is noted that their root nodes are the same, and are obtained from the reference graph G_{c1} by removing all the edges in the cut set $Q1 = \{ \langle M_2, V_1 \rangle, \langle M_3, V_1 \rangle \}$, in which the

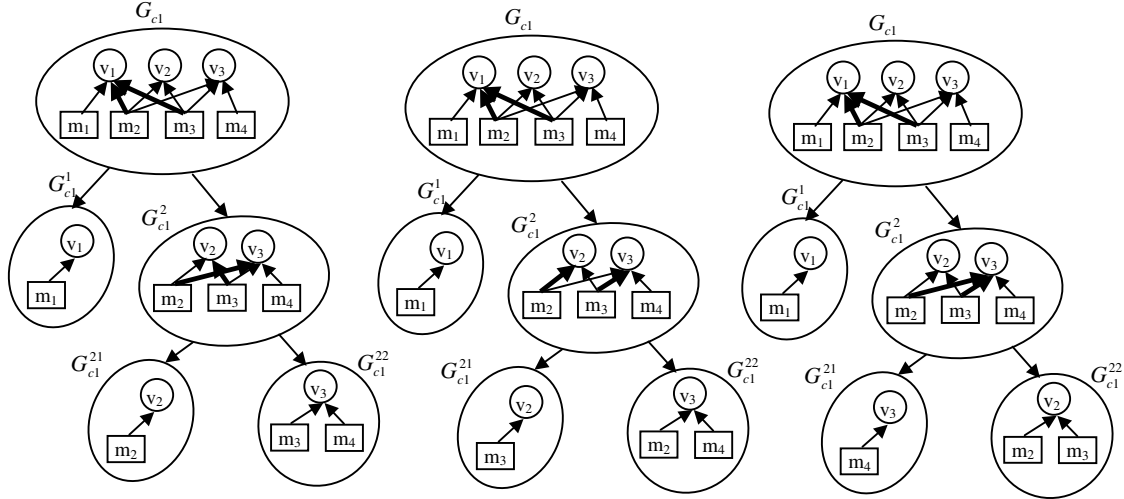


Figure 4 The structure tree of G_{c1} corresponding to $Q1$

cut set consists of edges with bold line. Based on these definitions, the cohesion for a class can be redefined as:

Definition 4 Class cohesion The cohesion for the reference graph $G_c = (N_c, E_c)$ of a class c is defined as $ICBMC(G_c)$

$$= \text{Max} \left\{ \frac{|M_g|}{|N_m(G(c))| \times |N_v(G(c))|} \times \frac{1}{2} \times \sum_{i=1}^2 ICBMC(G_{M_g}^i) \right\}$$

$Q \in \text{CS}(G)$

Compared with the original definition of Chae's cohesion measure, it is easy to know that the improved definition defines the connectivity factor and the structure factor as follows, respectively.

$$F_c(G) = \frac{|Q|}{|N_m(G_c)| \times |N_v(G_c)|}$$

$$F_s(G) = \frac{1}{2} \times [ICBMC(G_{cQ}^1) + ICBMC(G_{cQ}^2)]$$

Above formulae show that the connectivity factor is the ratio of the number of edges in a cut set to the number of all possible interaction edges and the structure factor is the average of two components of the reference of a class.

It is clear that the abnormal behavior in Chae's original definition have been corrected in the improved version of class cohesion. For example, consider the cohesion value of class $c1$, $c2$, $c3$, $c4$ and $c5$ shown in figure 2. Because the four structure trees corresponding to the reference graph G_{c1} are isomorphous, its cohesion value only needs to be calculated according to one of these structure trees.

Therefore

$$\begin{aligned} ICBMC(G_{c1}) &= \frac{|Q1|}{|N_m(G(c1))| \times |N_v(G(c1))|} \times \frac{1}{2} \times \sum_{i=1}^2 ICBMC(G_{Q1}^i) \\ &= \frac{2}{12} \times \frac{1}{2} \times [ICBMC(G_{c1}^1) + ICBMC(G_{c1}^2)] \\ &= \frac{2}{12} \times \frac{1}{2} \times [1 + \frac{2}{6} \times \frac{1}{2} \times (1+1)] = 1/9 \end{aligned}$$

The cohesion of class $c2$, $c3$ and $c4$ can be calculated similarly: $ICBMC(G_{c2}) = 1/6$, $ICBMC(G_{c3}) = 2/9$, $ICBMC(G_{c4}) = 5/96$, $ICBMC(G_{c5}) = 1/18$. Therefore, according to the improved Chae's measure, it is easy to conclude that

$$ICBMC(G_{c3}) > ICBMC(G_{c2}) > ICBMC(G_{c1}) > ICBMC(G_{c5}) > ICBMC(G_{c4})$$

Obviously, the above conclusion is consistent with our intuition.

4. Comparison with related works

In section 2, we state that the more complex the interactions among the members of a class are, the more cohesive the class is. This statement actually implies that a well-defined class cohesion metric should satisfy two constraints. On the one hand, for any class c and c' , if $R_c \subset R_{c'}$ (where R_c and $R_{c'}$ denote the set of relationships within class c and c' respectively), then the cohesion of class c' should be greater than that of c , i.e., $\text{Cohesion}(c) < \text{Cohesion}(c')$. This constraint is similar to the monotonic property Briand proposed, but stricter than it. Obviously, this constraint is reasonable. For example, let us consider the reference graphs of class $c1$, $c2$ and $c3$ shown in figure 2 again.

It is clear that $c2$ can be formed by adding an edge to $c1$ and $c3$ can be formed by adding an edge to $c2$, i.e., the interactions among the members of class $c2$ are more complex than that of class $c1$ and simpler than that of class $c3$ ($R_{c1} \subset R_{c2} \subset R_{c3}$). Hence, the cohesion of class $c2$ should be greater than that of class $c1$ and less than that of class $c3$ in intuition ($\text{Cohesion}(c1) < \text{Cohesion}(c2) < \text{Cohesion}(c3)$). However, $ICBMC$ satisfies this constraint but the original one does not.

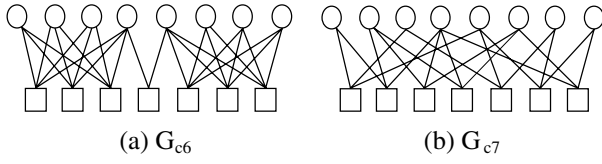


Figure 5 The reference graphs for class c6 and

On the other hand, the cohesion of a class is closely related to the distribution of the interactions among its member. The more divergent the distribution of the interactions among the members of a class is, the more cohesive the class is. The rationale behind this constraint is that the reference graph of a class with divergent interactions is more difficult to be understood and maintained. For instance, consider the reference graphs of class c6 and class c7 shown in figure 5. Obviously, Class c6 consists of two collections with little relationship and the members of class c7 relatively have relations among them, i.e., the distribution of the interactions of G_{c7} is more divergent than that of $G_{c6}^{[10]}$. Therefore, class c7 is more cohesive than class c6 in intuition. We convince that a class cohesion metric should satisfy these two constraints; otherwise it is not well defined.

As a consequence, when used to measure the cohesion of class c6 and class c7, the improved CBMC will also draw a conclusion consistent with intuition. To demonstrate this fact, let us examine the computation of the cohesion of class c6 and class c7. Obviously, when decomposed by cut set, G_{c6} corresponds to two isomorphic structure trees, one of which is shown in figure 6(a). However, G_{c7} corresponds to a lot of structure trees, one of which is shown in figure 6(b).

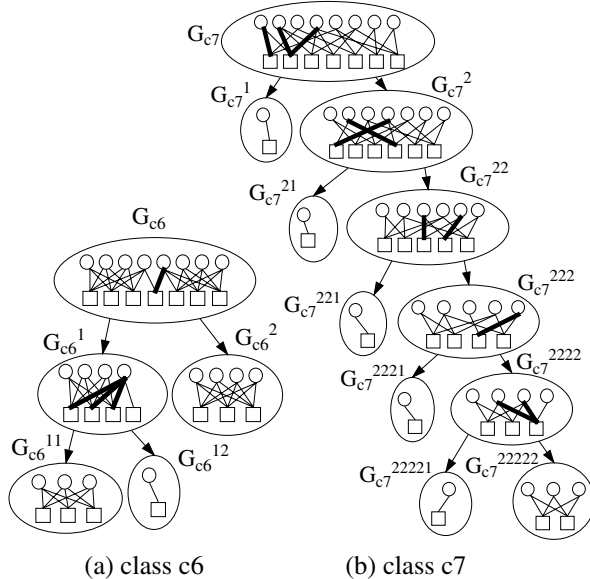


Figure 6 The structure trees of G_{c6} and G_{c7} corresponding to cut set

According to ICBMC, the class cohesion of class c6 can be computed as follows:

$$\begin{aligned} \text{ICBMC}(G_{c6}) &= F_c(G_{c6}) \times F_s(G_{c6}) \\ &= F_c(G_{c6}) \times \frac{1}{2} \times [\text{ICBMC}(G_{c6}^1) + \text{ICBMC}(G_{c6}^2)] \\ &= \frac{1}{56} \times \frac{1}{2} \times \{F_c(G_{c6}^1) \times \frac{1}{2} \\ &\quad \times [\text{ICBMC}(G_{c6}^{11}) + \text{ICBMC}(G_{c6}^{12})] + 1\} \\ &= \frac{1}{56} \times \frac{1}{2} \times [\frac{3}{16} \times \frac{1}{2} \times (1+1) + 1] = 0.0106 \end{aligned}$$

For simplifying the computation, let $\text{ICBMC}^{T1}(G_{c7})$ represent the cohesion value of class c7 corresponding to the structure tree T_i . If we name the structure tree shown in figure 6(b) T_1 , then it is easy to conclude that

$$\begin{aligned} \text{ICBMC}^{T1}(G_{c7}) &= F_c^{T1}(G_{c7}) \times F_s^{T1}(G_{c7}) \\ &= F_c^{T1}(G_{c7}) \times \frac{1}{2} \times [\text{ICBMC}^{T1}(G_{c7}^1) + \text{ICBMC}^{T1}(G_{c7}^2)] \\ &= \frac{3}{56} \times \frac{1}{2} \times \{1 + F_c^{T1}(G_{c7}^2) \times \frac{1}{2} \\ &\quad \times [\text{ICBMC}^{T1}(G_{c7}^{21}) + \text{ICBMC}^{T1}(G_{c7}^{22})]\} \\ &= \frac{3}{112} \times \{1 + \frac{1}{42} \times \{1 + F_c^{T1}(G_{c7}^{22}) \times \\ &\quad \frac{1}{2} \times [\text{ICBMC}^{T1}(G_{c7}^{221}) + \text{ICBMC}^{T1}(G_{c7}^{222})]\}\} \\ &= \frac{3}{112} \times \{1 + \frac{1}{42} \times \{1 + \frac{1}{30} \times \{1 + F_c^{T1}(G_{c7}^{222}) \times \\ &\quad \frac{1}{2} \times [\text{ICBMC}^{T1}(G_{c7}^{2221}) + \text{ICBMC}^{T1}(G_{c7}^{2222})]\}\}\} \\ &= \frac{3}{112} \times \{1 + \frac{1}{42} \times \{1 + \frac{1}{30} \times \{1 + \frac{1}{40} \times [1 + \frac{2}{12} \times \frac{1}{2} \times (1+1)]\}\}\} \\ &= 0.0274 \end{aligned}$$

According to the definition of ICBMC, $\text{ICBMC}(G_{c7}) \geq \text{ICBMC}^{T1}(G_{c7})$, i.e., $\text{ICBMC}(G_{c7}) \geq 0.0274$. Therefore, a conclusion that the measuring value of the cohesion of class c7 is greater than that of class c6 can be drawn. This result is consistent with our intuition. In other words, ICBMC satisfies the second constraint.

To make the difference between ICBMC and CBMC more explicit, let us consider the cohesion value of class c6 and class c7 computed by CBMC. Apparently, when decomposed by the set of glue methods, G_{c6} corresponds to a single structure tree shown in figure 7(a). However, G_{c7} corresponds to a lot of structure trees, one of which has maximum cohesion value shown in figure 7(b).

According to the definition of CBMC, it is not difficult to conclude that $\text{CBMC}(G_{c6}) = 0.1429$ and $\text{CBMC}(G_{c7}) = 0.1786$. If the relation of cohesion value between class c6 and class c7 is carefully examined, an interesting result will be discovered. When measured by ICBMC, the cohesion value of class c7 is twice over than that of class c6. However, when measured by the original one, the cohesion value of class c7 is very closed to that of class c6. In other words, although both of CBMC and ICBMC satisfy the second constraint, the distribution of the interactions among the members of a class is well characterized by the latter other than by the former.

The cohesion values of class c1, c2, c3, c4, c5, c6 and c7 measured by five cohesion measures are summarized in table 1. According to LCOM, all of these seven classes

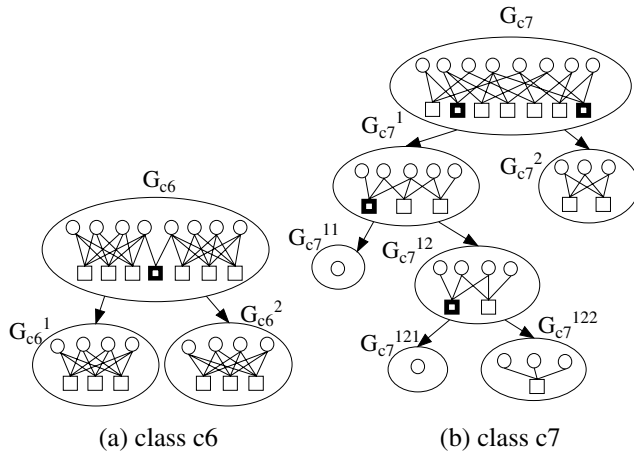


Figure 7 The structure trees of G_{c6} and G_{c7} corresponding to glue methods

have the same cohesion. Therefore, LCOM has little discriminating power. For these seven classes, it is easy to see that RCI and OL_2 have the same discriminating power. However, when used to measure class c6 and c7, both of RCI and OL_2 draw an incorrect conclusion. When applied to measure class c1, c2 and c3, CBMC draws a conclusion that is not consistent with our intuition. Based on these discussions and analyses, it is easy to know that ICBMC satisfies both of these two constraints above-mentioned but these typical cohesion measures do not. Therefore, it is reasonable to convince that ICBMC is a well-defined metric but these typical cohesion measures are not.

	LCOM	RCI	OL_2	CBMC	ICBMC
c1	0	0.6667	0.3750	0.5000	0.1111
c2	0	0.7500	0.5000	0.3750	0.1667
c3	0	0.8333	0.6389	0.5000	0.2222
c4	0	0.5000	0.1597	0.1875	0.0521
c5	0	0.5833	0.2500	0.1875	0.0556
c6	0	0.4643	0.1046	0.1429	0.0106
c7	0	0.3750	0.0392	0.1786	≥ 0.0274

Table 1 Measuring results

5. An application to classes restructuring

Generally speaking, the relatedness among the class members of a well-designedness class should be tight; otherwise it might imply that the attributes and methods corresponding to several different concepts are grouped into the class, i.e., the class has some design deficiency. ICBMC is just used to quantify the relatedness among the members of a class and hence can provide developers with feedback on design quality of the class. High cohesion value of a class measured by ICBMC means that the relation among its members is tight. On the contrary, low cohesion value of a class measured by ICBMC states that the relation among its members is low. It is clear that

ICBMC can characterize the design quality of a class.

Based on this rationale, we believe that ICBMC could be used to restructure classes that have design deficiency. For example, consider the class *Car1* with seven attributes shown in figure 10(a). It has a constructor *Car1*, four normal methods *RotateTires*, *RepalceTires*, *RepairEngine* and *Drive*, two delegation methods *TireMiles* and *MaxTireMiles*, and three access methods *EngineMiles*, *MaxEngineMiles* and *GetCarMiles*. The corresponding reference graph for class *Car1* is shown in figure 8. It is noted that all special methods such as constructor, delegation methods and access method have been excluded in G_{Car1} .

Clearly, the set of all cut set of G_{Car1} consists of two elements, i.e., $CS(G_{Car1}) = \{\{<Drive, engine_miles>\}, \{<RepairEngine, engine_miles>\}\}$. Specially, when the edge $<Drive, engine_miles>$ is removed, G_{Car1} will be decomposed into two components. One has one normal method *RepairEngine* and two attributes *engine_miles* and *engine_max_miles*. The other has three normal methods *Drive*, *RotateTires*, *ReplaceTires* and five attributes *lf_tire*, *lb_tire*, *rf_tire*, *rb_tire* and *miles*. At the same time, the normal method *Drive* also accesses the attributes in the first component. As a result, class *CEngine* is created for the first component and class *Car2* created for the second component has an instance variable *engine* of class *CEngine* and preserves the interface of class *Car1* (figure 10(b)). The corresponding reference graphs for *CEngine* and *Car2* are depicted in figure 9(a) and figure 9(b), respectively.

In fact, there exist three important relationships among objects in an object-oriented system, i.e., *is-kind-of*, *is-analogous-to* and *is-part-of* relationships. If objects of A are a kind of class B, then A could be represented as a subclass of B. If objects of class A are analogous to objects of class B, then a new common abstract class could be defined to capture their commonalities. If an object of class A is a part of an object of class B, then A could be defined as a component class of B, i.e., class B is a composite class that has instance variables of component class A. However, the relationships among objects in many real object-oriented programs are sometimes not clear and even incorrectly used. Therefore, many methods such as refactoring technique, concept lattice based inheritance hierarchy restructuring technique etc. are developed to cope with this problem.

Actually, the restructuring of poorly designed classes using class cohesion is an effective technique that is used to make the *is-part-of* relationships among objects explicit. Clearly, the relationship between a car and its engine is a whole-part relationship. However, class *Car1* shown in figure 10(a) is not a well-designed class that captures the whole-part relationship explicitly. When class *Car1* is restructured using the improved CBMC, a new independent class *CEngine* is generated and represented as

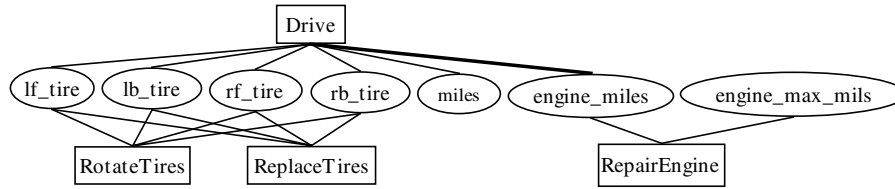


Figure 8. Reference graph for class Car1 (G_{Car1})

a component class of composite class Car2, which is consistent with our intuition. As a result, two statements used to initialize instance variables `engine_miles` and `engine_max_miles` in `Car1::Car1()` are extracted and encapsulated as `CEngine::CEngine()`. A statement used to modify `engine_miles` in `Car1::Drive()` is extracted and encapsulated as a normal method `CEngine::Drive()`. In addition, class `CEngine` provides two access methods `GetEngineMiles` and `GetEngineMaxMiles`. It is noted that restructuring is behavior preserving, i.e., restructuring should not change the behavior of old class. Therefore, the method `RepairEngine` in class `Car2` is preserved by invoking the corresponding method in class `CEngine` to provide the same interface as class `Car1`. On the other hand, the direct reference to instance variable `engine_miles` and `engine_max_miles` in the methods `Drive`, `EngineMiles` and `MaxEngineMiles` is converted to the invocation of the corresponding normal or access methods in class `CEngine`.

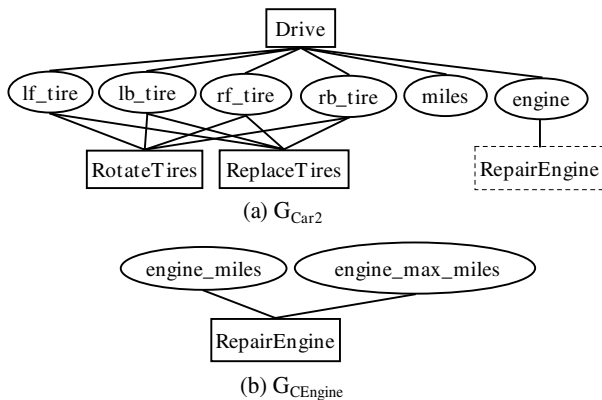


Figure 9. Reference graph for class Car2 and the new class CEngine

6. Conclusions

In this paper, we analyze and discuss the problems in CBMC and propose an improved version ICBMC. It is clear that the improved definition of class cohesion measure is consistent with our intuition, which overcomes the limitations of not only CBMC, but also previous class cohesion measures. In fact, ICBMC characterizes interaction pattern of a reference graph better than the original one does. On the other hand, it is easy to know that ICBMC could be used to be a guideline about quality

evaluation and restructure poorly designed classes. Therefore, we believe that ICBMC is a well-defined class cohesion metric, which overcomes the limitations of CBMC and extends its application. In the future work, we expect to further improve the ability to represent the patterns of the interactions using information theory^[12], develop a tool to automate our cohesion and then apply it to real commercial software, evaluate the design quality of classes, extract design patterns from the legacy systems and test object-oriented software based on our previous work^[13-14].

References

- [1] W. P. Stevens, G. J. Myers, and L.L. Constantine. Structured design. IBM Systems Journal, 1974, 13(2): 115-139.
- [2] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, 1994, 20(6): 476-493.
- [3] M. Hitz and B. Montazeri. Measuring Coupling and Cohesion in Object-Oriented Systems. In: Proceedings of International Symposium on Applied Corporate Computing, Monterrey, Mexico, October 1995: 25-27.
- [4] B. Henderson-Sellers. Software Metrics, Prentice Hall, Hemel Hempstead, U.K., 1996
- [5] J. L. Bansiya and Etzkorn et al. A class cohesion metric for object oriented designs. Journal of Object-oriented Programming, 1999, 11(8): 47-52.
- [6] H. S. Chae and Y. R. Kwon. A Cohesion Measure for Classes in Object-Oriented Systems. In: Proceedings of the 5th International Software Metrics Symposium, IEEE Computer Society Press, Bethesda, USA, 1998: 158-166.
- [7] H. S. Chae, Y. R. Kwon and D. H. Bae. A Cohesion Measure for Object-Oriented Classes. Software Practice & Experience, 2000, 30(12): 1405-1431.
- [8] L. C. Briand J. W. Daly and J. Wüst. A Unified Framework for Cohesion Measurement in Object-Oriented Systems, IESE-Report 040.97/E, December 1997.
- [9] Baowen Xu and Yuming Zhou. Comments on 'A Cohesion Measure for Object-oriented Classes' by Heung Seok Chae, Yong Rae Kwon and Doo Hwan Base. Software Practice & Experience, 2001, 31(14): 1381-1388.
- [10] H. S. Chae, Y. R. Kwon and D. H. Bae. Response to 'Comments on: A Cohesion Measure for Object-oriented Classes'. Software Practice & Experience, 2001, 31(14): 1389-1392.
- [11] Xuefei Yang. Research on cohesion measures for classes. Master Thesis, Department of Computer Science & Engineering in Southeast University, March, 2002: 33-36.
- [12] C. E. Shannon. A Mathematical Theory of Communications.

```
#include <iostream>
using namespace std;
class CTire
{
private:
    int miles, max_miles;
public:
    void Drive(int m) {miles += m;}
    int TireMiles(){return miles;}
    int MaxTireMiles(){return max_miles;}
    CTire() {max_miles = 40000;
            miles = 0;}
};
class Car1
{
private:
    CTire * lf_tire, * lb_tire, * rf_tire, * rb_tire;
    int engine_miles, engine_max_miles;
    int miles;
public:
    Car1(){
        lf_tire = new CTire; lb_tire = new CTire;
        rf_tire = new CTire; rb_tire = new CTire;
        engine_max_miles = 100000;
        engine_miles = 0; miles = 0;
    }
    void RotateTires() {
        // exchange lf_tire and lb_tire
        // exchange rf_tire and rb_tire
    }
    void ReplaceTires(CTire * lf, CTire * rf,
                    CTire * lb, CTire * rb) {
        // replace lf_tire, lb_tire, rf_tire, rb_tire
        // with lf, lb, rf, rb respectively
    }
    void RepairEngine(){
        engine_miles = 0;
        engine_max_miles += 10000;
    }
    void Drive(int m){
        miles += m;
        engine_miles += m;
        lf_tire->Drive(m); lb_tire->Drive(m);
        rf_tire->Drive(m); rb_tire->Drive(m);
    }
    int TireMiles(){return lf_tire->TireMiles(); }
    int MaxTireMiles()
        {return lf_tire->MaxTireMiles();}
    int EngineMiles(){return engine_miles;}
    int MaxEngineMiles()
        {return engine_max_miles; }
    int GetCarMiles(){return miles;}
};
```

(a) class Car1

```
#include <iostream>
using namespace std;
class CTire
{ // remains no change};
class CEngine
{
private:
    int engine_miles, engine_max_miles;
public:
    int GetEngineMiles(){return engine_miles;}
    int GetEngineMaxMiles()
        {return engine_max_miles;}
    void RepairEngine (){
        engine_miles = 0;
        engine_max_miles += 10000; }
    void Drive(int m){engine_miles+=m;}
    CEngine(){
        engine_miles = 0;
        engine_max_miles = 100000;
    }
};
class Car2
{
private:
    CTire * lf_tire, * lb_tire, * rf_tire, * rb_tire;
    CEngine engine;
    int miles;
public:
    Car2(){
        lf_tire = new CTire; lb_tire = new CTire;
        rf_tire = new CTire; rb_tire = new CTire;
        miles = 0;
    }
    void RotateTires() { // same as Car1::RotateTires }
    void ReplaceTires(CTire * lf, CTire * rf,
                    CTire * lb, CTire * rb)
        { // same as Car1::RotateTires }
    void RepairEngine(){engine.RepairEngine();}
    void Drive(int m){
        miles += m;
        engine.Drive(m);
        lf_tire->Drive(m); lb_tire->Drive(m);
        rf_tire->Drive(m); rb_tire->Drive(m);
    }
    int TireMiles(){return lf_tire->TireMiles(); }
    int MaxTireMiles()
        {return lf_tire->MaxTireMiles();}
    int EngineMiles()
        {return engine.GetEngineMiles();}
    int MaxEngineMiles()
        { return engine.GetEngineMaxMiles(); }
    int GetCarMiles(){return miles;}
};
```

(b) class Car2

Figure 10 Program fragment before and after restructuring